Willkommen zum Informix Newsletter

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

der Sommer war aussergewöhnlich sonnig und warm. Auch die Redaktion hat diese Zeit genutzt, um Energie zu tanken für die dunklere und kältere Jahreszeit.
Den Sommer über hatten wir zahlreiche Anfragen zu Hilfsfunktionen wie z.B. isdate(), welche im SQL Umfang anderer Datenbanken angeboten wird. Bei INFORMIX lässt sich dafür stets eine einfache Prozedur bauen, die anschliessend überall genutzt werden kann.

Mit neuem Schwung und vielen Ideen starten wir nun in das vierte Quartal und sammeln die Themen, damit wir kurz vor Weihnachten wieder eine interessante und informative Ausgabe erstellen können.

Ihr TechTeam

TechTipp: Optionen des ONMODE (onmode -B reset) Reset Bufferpool

Mit jedem SQL Statement wird auf Daten im Cache (BUFFERPOOL) zugegriffen.
Sind die Daten noch nicht im BUFFERPOOL enthalten, so werden diese von den Platten (Chunks) eingelesen. Bei jedem Lesen von Platte werden zusätzliche Daten, je nach Einstellung des Konfigurationsparameters READAHEAD, übertragen mit der Hoffnung, dass diese bei einem der nachfolgenden Statements genutzt werden.

Spätestens bei einem Checkpoint werden alle veränderten DatenPages aus dem BUFFERPOOL zurück auf die Platte geschrieben.

Um dieses Ausschreiben unabhängig von einem Checkpoint zu ermöglichen, existiert die Option „onmode -B“. Mit Version 14.10.xC2 wurde nun eine weitere Option „reset“ hinzugefügt, die es z.B. beim Testen ermöglicht den Cache zu leeren um identische Voraussetzungen zu schaffen, ohne dass die Instanz neu gestartet werden muss.

Der Aufruf wird im „online.log“ protokolliert:

```
2022-08-02 18:43:43.701  All dirty buffers flushed to disk.
```

Soll zudem der Inhalt des BUFFERPOOLS gelöscht werden, so kann dies mittels „onmode -B reset“ erfolgen.

```
2022-08-02 18:43:52.644  All dirty buffers flushed to disk.
                         All cached buffers invalidated.
```

Dies macht z.B. dann Sinn, wenn bei einer nächtlichen Verarbeitung ein Teilbereich abgeschlossen ist, und im nächsten Schritt Daten ausschliesslich aus anderen Tabellen und Indexen benötigt werden.

TechTipp: Optionen des ONMODE (onmode -h) Refresh sqlhosts cache

Die INFORMIX Instanz besitzt einen Cache für die Informationen aus den Dateien „hosts“, „services“, „user“, „group“ und „sqlhosts“.

Diese werden, je nach Werten im Konfigurationsparameter NS_CACHE, in bestimmten Intervallen aktualisiert, falls sie sich geändert haben.

Um einen sofortigen Refresh der Daten aus der Konfigurationsdatei „sqlhosts“ zu erzwingen, wurde die Option „onmode -h“ eingeführt, die bei Änderung der Datei sqlhosts die Daten der Informix Verbindungen neu in den Cache lädt.

Diese Aktualisierung der Daten kann erzwungen werden, auch wenn sich das Änderungsdatum der Datei nicht geändert hat, indem als weiterer Parameter „force“ angegeben wird:

```
„onmode -h force“
```

TechTipp: Optionen des ONMODE (onmode -a) Add Shared Memory

Im laufenden Betrieb nutzt der Informix Server den Virtuellen Shared Memory z.B. um Zwischenergebnisse abzulegen, oder um Daten zu sortieren.
Hierbei wird, falls der Platz im initialen Virtuellen Memory Segment für die aktuellen Aufgaben nicht ausreicht, weiterer Platz durch zusätzliche Segmente in der Grösse SHMADD geschaffen.

Mit der Option

„onmode -a <segment_size>"

besteht die Möglichkeit, kurzfristig vor einer Verarbeitung ein zusätzliches Memory Segment hinzuzufügen, dessen Grösse auf die zu erwartende Last angepasst ist.

Wenn die Ausführung der Applikation abgeschlossen ist, und die Session ihre Verbindung zur Datenbank beendet hat, empfehlen wir einen „onmode -F" aufzurufen, der im Memory aufräumt. Hierbei werden nicht mehr benötigte Strukturen gelöscht und nicht mehr belegte Virtuelle Memory Segmente an das Betriebssystem zurück gibt.

TechTipp: SPL: ISDATE() - Abfrage auf ein gültiges Datum

Immer wieder erreicht uns die Anfrage, ob INFORMIX auch eine Funktion ISDATE() besitzt, mit der man überprüfen kann, ob ein Wert ein gültiges Datum beinhaltet. Je nach Datenbankhersteller gibt es hier unterschiedliche SQL Varianten, die diese Prüfung bewerkstelligen.

Die Lösung bei INFORMIX ist ganz einfach:
Wir erstellen uns eine Prozedur, die genau diese Eigenschaft überprüft.

```
create procedure if not exists isdate(x_in varchar(25))
returning int as isdate
    define ret int;
    define test_date date;

    on exception
        let ret = 0;
        return ret;
    end exception with resume;

    let ret = 0;
    let test_date = x_in::date;
    let ret = 1;
    return ret;
end procedure;
```

Test: `execute procedure isdate('23.02.2022');`

Hinweis: Das Datumsformat ist abhängig von den Umgebungsvariablen DBDATE, CLIENT_LOCALE und GL_DATE.

TechTipp: SPL: ISDATETIME() - Abfrage auf einen gültigen Zeitstempel

Analog der Überprüfung, ob ein Wert ein gültiges Datum enthält, lassen sich auch weitere Prüfungen implementieren. So zum Beispiel, ob es sich beim Wert um einen gültigen Eintrag für den Datentypen DATETIME handelt:

```
create procedure if not exists isdatetime(x_in varchar(25))
returning int as isdatetime
    define ret int;
    define test_date datetime year to second;

    on exception
        let ret = 0;
        return ret;
    end exception with resume;

    let ret = 0;
    let test_date = x_in::datetime year to second;
    let ret = 1;

    return ret;
end procedure;

Test: execute procedure isdatetime('2022-02-23 12:42:23');
```

Hinweis: Das Datumsformat ist abhängig von den Umgebungsvariablen DBTIME, CLIENT_LOCALE und GL_DATETIME.

Das Vorgehen ist bei diesen Prüfungen immer identisch:
Man versucht die eingegebene Zeichenfolge mittels „Cast“ in den Datentypen zu konvertieren, der geprüft werden soll.
Schlägt dieser Cast fehl, so handelt es sich bei der Eingabe wahrscheinlich nicht um einen Wert, der umgewandelt werden kann.
Hiermit könnte z.B. auch abgefragt werden, ob eine Eingabe numerisch ist.
Die Anwendungsgebiete sind vielfältig und flexibel.

TechTipp: SPL: Replace ASCII – Sonderzeichen ersetzen

Uns hat eine Anfrage erreicht, wie man aus einem übernommenen Text (varchar) Zeichen entfernen kann, die bei der Darstellung Probleme bereiten. Im konkreten Fall handelte es sich um ASCII code 26 = SUB (Substitute), ASCII code 27 = ESC (Escape), ASCII code 28 = FS (File separator), ASCII code 29 = GS (Group separator), ASCII code 30 = RS (Record separator).
Die implizit vorhandene Funktion „replace“ nimmt leider diese Zeichen nicht als Argument an, so dass wir hier eine besondere Lösung finden mussten.

Mit Hilfe einer Prozedur konnten wir eine Lösung bereitstellen, die einfach und zudem flexibel ist. Hierbei nutzen wir die Funktion ASCII um herauszufinden, ob das gesuchte (und zu ersetzende) Zeichen im String vorhanden ist.
Einzige Voraussetzung: Der ASCII-Wert des zu ersetzenden Zeichens muss bekannt sein.

Hier unsere Lösung:

```
drop procedure if exists replace_ascii_val
    (varchar(100),int, char(1));

create procedure if not exists replace_ascii_val(
    in_string varchar(100),
    ascii_val int,
    target_val char(1)
)
returning varchar(100)

define i int;
define x_char nchar(1);
define result varchar(200);
let i = 1;
let result = '';
while (i <= len(in_string))
    begin
     let x_char = CHR(ascii(substr(in_string,i,1)));
        if ascii(x_char) = ascii_val
        then let result = result||target_val;
        else let result = result||x_char;
        end if;
     let i = i+1;
    end
end while
return result;
end procedure;
```

Test:

```
execute procedure replace_ascii_val("Alles wird gut",32,"*");
```

Ergebnis:

```
Alles*wird*gut
```

TechTipp: Task (file info) - Informationen über Dateien abfragen

Abfragen innerhalb der Datenbank können mittels SQL durchgeführt werden. Nun gibt es Situationen, bei denen man Informationen über Dateien benötigt, z.B. ob die Datei existiert, oder wann diese zuletzt geändert wurde.

Informix bietet hierzu die Möglichkeit mittels eines Tasks die verfügbaren Informationen zu einer Datei im SQL zu erhalten.
Der Task „file status“ bzw. „print file info“ aus der Datenbank sysadmin kann hierfür genutzt werden.

Die folgenden Beispiele zeigen, wie dieser Task genutzt werden kann.

1. Abfrage, ob eine Datei vorhanden ist (Datei „check_ptnhdr.xxx“ existiert nicht):

```
execute function sysadmin:task(
    "file status",
    "/home/informix/HEALTH/LOAD/check_ptnhdr.xxx"
);
```

```
Ergebnis:
(expression)    File name   =
/home/informix/HEALTH/LOAD/check_ptnhdr.xxx
            Error       = 2
            Message     = File does not exist
```

2. Abfrage der Informationen zu einer bestehenden Datei „check_ptnhdr.unl“:

```
execute function task(
    "print file info",
    "/home/informix/HEALTH/LOAD/check_ptnhdr.unl"
);
```

```
Ergebnis:
(expression)    File name   =
/home/informix/HEALTH/LOAD/check_ptnhdr.unl
            Is File     = 1
            Is Directory    = 0
            Is Raw Device   = 0
            Is Block Device = 0
            Is Pipe     = 0
            File Size   = 690971
            Last Access Time   = 09/08/2022 16:39:48
            Last Modified Time = 09/08/2022 16:39:47
            Status Change Time = 09/08/2022 16:39:47
            User Id            = 1001
            Group id           = 1001
            File Flags         = 33204
```

Es werden neben den Informationen, ob die Datei existiert, auch deren Eigenschaften zurückgegeben.
Diese sind

- ob es sich um eine Datei handelt
- ob es ein Verzeichnis ist
- ob es sich um ein RawDevice handelt
- ob es ein Blockdevice ist
- ob es eine Pipe ist.

Zudem finden sich die Informationen zur User-ID und Group-ID wieder, die Eigentümer der Datei sind.

Weiters finden sich Informationen zum letzten Zugriff und zum Änderungsdatum der Datei.

All diese Informationen werden als eine, zusammenhängende Information zurückgegeben. Eine Lösung für die Aufgabe, diese Information für SQL abfragbar zu zerlegen findet sich in einem folgenden Artikel im Newsletter.

TechTipp: Task (delete file) - Datei einlesen und löschen (Beispiel)

Eine praktische Anwendung ist zum Beispiel zu prüfen, ob eine Datei existiert, bevor versucht wird, deren Inhalt zu laden. Es könnte auch das letzte Änderungsdatum abgefragt werden, um zu sehen, ob neue Daten angekommen sind.
Ist die Datei erfolgreich eingelesen, so kann diese mittels SQL gelöscht werden.

Beispiel:

```
drop table if exists file_status;
select sysadmin:task(
    "print file info","/transfer/check_ptnhdr.unl"
    )::lvarchar(1000) as message
from systables where tabid = 1
into temp file_status with no log;

select
        case
                when message like "%not exist%" then "DATEI NICHT DA"
                else "OK"
        end
from file_status;
```

Nach erfolgreichem Laden der Daten kann die Transferdatei gelöscht werden:

```
select sysadmin:task(
    “delete file“,“/transfer/check_ptnhdr.unl“
)
from systables where tabid = 1;
```

TechTipp: Zerlegen der Rückgabe des Tasks „file status“

Die Information, die als eine lange Zeichenkette vom Task zurückgegeben wird, ist aus allen Einzelinformationen zusammengesetzt und enthält zudem Zeilenumbrüche zur Formatierung.
Daher ist daher Sinnvoll, diese Informationen zu zerlegen, und in einer mittels SQL abfragbaren Tabelle abzulegen.

Anmerkung: Dies ist ein erster, schneller Versuch der Zerlegung.
Bessere Implementierungen sind als Rückmeldung herzlich willkommen und finden sicher einen Platz in der Ausgabe Q4 des Newsletters.

Schritt 1: Wir schaffen uns erst einmal eine Tabelle, in der wir die zusammengehörigen Informationen eintragen können.

```
drop table if exists tmp_message;
create temp table tmp_message (
        txt     lvarchar(1000)
);
```

Schritt 2: Wir zerlegen die Zeichenkette in die Informationsblöcke, indem wir beim Zeichen „\n“, das den ASCII-Wert 10 hat, die Zeichenkette zerlegen. Hierfür nutzen wir eine selbst geschriebene Funktion split_string_ascii(), die im folgenden Artikel vorgestellt wird:

```
insert into tmp_message
execute procedure split_string_ascii(
    sysadmin:task(
        "file status",
        "/home/informix/TECH_NL/2022/2022_Q3/TEST.txt"
    ),10    <--- ASCII-Wert zur Zerlegung als Argument
);

drop table if exists tmp_fileinfo;
create temp table tmp_fileinfo
    (name lvarchar(1000), value lvarchar(1000));
```

Schritt 3: Nun müssen wir noch jede Kombination aus Name und Wert teilen, indem wir beim Zeichen „=“ die Information zerlegen. Hierfür nutzen wir eine selbst geschrieben Funktion split_string(), die im folgenden Artikel vorgestellt wird:

```
insert into tmp_fileinfo
select split_string(txt,'=',1), split_string(txt,'=',2)
from tmp_message;
```

Ergebnis: Eine Tabelle, in der wir gezielt die Abfragen ausführen können:

```
select name, value
from tmp_fileinfo
```

TechTipp: SPL: split_string()

Die Stored Procedure zerlegt eine Zeichenkette in zwei Teile, wobei als Trenner der übergebene Characterwert dient. Der dritte Parameter bestimmt, ob Teil1 oder Teil2 zurückgegeben werden:

```
drop procedure if exists split_string(lvarchar(1000),char(1),int);
create procedure "informix".split_string (
    in_str lvarchar(1000),
    in_val char(1),
    nr int
    )
    returning lvarchar(1000) as part
    define out_str1 lvarchar(1000);
    define out_str2 lvarchar(1000);
    define i,k,n int;

    let out_str1 = "";
    let out_str2 = "";
    let i = 1; let k = 0; let n = 0;

    while (i < length(in_str))
        if substr(in_str,i,1) = in_val
        then
            let n = k;
            let k = i;
            let out_str1 = substr(in_str,1,i-1);
            let out_str2 = substr(in_str,i+1,length(in_str));
            if nr = 1
                then return out_str1;
                else return out_str2;
            end if;
        end if;
        let i = i+1;
    end while;
end procedure;
```

TechTipp: SPL: split_sting_ascii()

Die Stored Procedure zerlegt eine Zeichenkette in Einzelteile, wobei als Trenner der übergebene ASCII-Wert des Trennzeichen dient:

```
drop procedure if exists split_string_ascii (lvarchar(1000),int);
create procedure split_string_ascii (
    in_str lvarchar(1000),
    in_val int
    )
    returning lvarchar(1000) as part
    define out_str lvarchar(1000);
    define txt lvarchar(1000);
    define i,k,n int;

    let out_str = "";
    let i = 1;
    let k = 0;
    let n = 0;

    while (i < length(in_str))
        if ascii(substr(in_str,i,1)) = in_val
        then
            let n = k+1;
            let k = i;
            return substr(in_str,n,i-n)with resume;
        end if;
        let i = i+1;
    end while;
    return substr(in_str,k+1,length(in_str)-k);
end procedure;
```

TechTipp: Isolation Level der aktuellen Session bestimmen

Wenn man in einer Applikation das Isolation Level zeitweise ändern will, um z.B. die aktuellen Lagerbestände abzufragen, ohne von offenen Transaktionen daran gehindert zu werden, danach aber im Programm mit dem bisherigen Isolation Level weiter machen will, so benötigt man die Information, in welchem Isolation Level sich die Applikation derzeit befindet.
Dies kann recht einfach mittels SQL abgefragt werden:

```
select
  case
     when o.odb_isolation = 1 then "dirty read"
     when o.odb_isolation = 2 then "commited read"
     when o.odb_isolation = 3 then "cursor stability"
     when o.odb_isolation = 5 then "repeatable read"
     when o.odb_isolation = 11 then "commited read last committed"
     else "error"
  end
from sysmaster:syssessions s, sysmaster:sysopendb o
where s.sid = o.odb_sessionid
and s.sid = DBINFO('sessionid')
and o.odb_dbname = DBINFO('dbname')
;
```

Nutzung des INFORMIX Newsletters

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit.
Die IUG hat sich dankenswerterweise dazu bereit erklärt, den INFORMIX Newsletter auf ihren Web Seiten zu veröffentlichen.
Da uns weder Tippfehler noch Irrtümer fremd sind, bitten wir hier um Nachsicht falls sich bei der Recherche einmal etwas eingeschlichen hat, was nicht wie beschrieben funktioniert.
Rückmeldungen hierzu sind herzlich Willkommen !
Die gefundenen Tippfehler dürfen zudem behalten und nach Belieben weiterverwendet werden.
Eine Weiterverbreitung in eigenem Namen (mit Nennung der Quelle) oder eine Bereitstellung auf der eigenen HomePage ist ausdrücklich erlaubt. Alle hier veröffentlichen Scripts stehen uneingeschränkt zur weiteren Verwendung zur Verfügung.

Die Autoren dieser Ausgabe

Andreas Legner
INFORMIX Advanced Support
HCL Technologies

Martin Fuerderer
INFORMIX Development
HCL Technologies

Gerd Kaluzinski DataOps Application Consultant
IBM Software
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Dank auch an die vielen Helfer im Hintergrund.

Nicht zu vergessen der Dank an die Informix User Group, ohne die es keinen Neuanfang des INFORMIX Newsletters gegeben hätte und die dankenswerter Weise die Verteilung übernimmt.

Foto Nachweis:
Kürbisplantage nahe Lindau im August 2022 (Gerd Kaluzinski)